

# Stanford CS224W: Label Propagation on Graphs

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# ANNOUNCEMENTS

- **This Thursday (2/2):** HW 1 due, HW 2 out
- **Next Tuesday (2/7):** Project proposal due
- **Next Thursday (2/9):** Colab 2 due, Colab 3 out

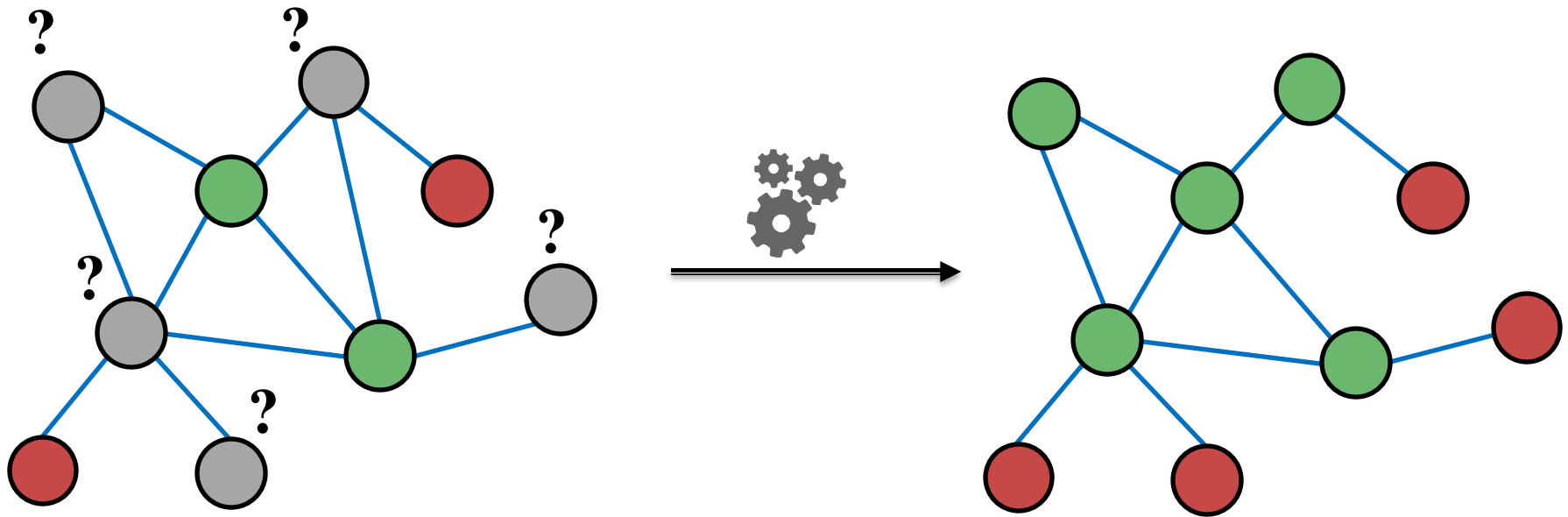
CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# Today's Lecture: Outline

- **Main question today:** Given a network with labels on some nodes, how do we **assign labels to all other nodes in the network?**
  - **Example:** In a network, some nodes are fraudsters, and some other nodes are fully trusted. **How do you find the other fraudsters and trustworthy nodes?**
- **Node embeddings** (from **random walks, GNN**) is a method to solve this problem
  - Are there alternative solutions based on the network topology?

# Setting: Node Classification



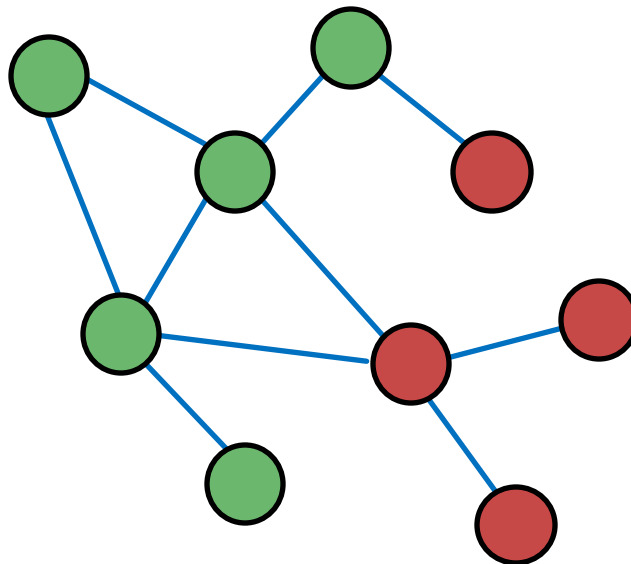
- Given labels of some nodes
- Let's predict labels of unlabeled nodes
- **Transductive node classification** (also called semi-supervised node classification)

# Today's Lecture: Outline

- **Main question today:** Given a network with labels on some nodes, how do we assign labels to all other nodes in the network?
- **Today we will discuss an alternative framework:** **Label propagation**
- **Intuition:** **Correlations** exist in networks.
  - Connected nodes tend to share the same label
- **We will look at three techniques today:**
  - **Label propagation**
  - **Correct & Smooth**
  - **Masked label prediction**

# Observation: Correlations in Networks

- Behaviors of nodes are **correlated** across the links of the network
- **Correlation**: Nearby nodes have the same color (belonging to the same class)

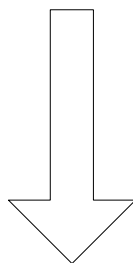


# Correlations Exist in Networks

- Two explanations for why behaviors of nodes in networks are correlated:

## Homophily

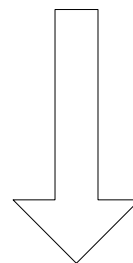
Individual  
Characteristics



Social  
Connections

## Influence

Social  
Connections

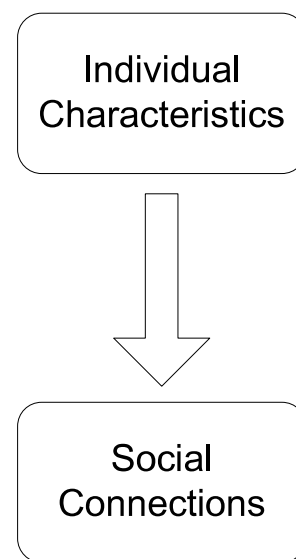


Individual  
Characteristics

# Social Homophily

- **Homophily**: The tendency of individuals to associate and bond with similar others
  - *“Birds of a feather flock together”*
  - It has been observed in a vast array of network studies, based on a variety of attributes (e.g., age, gender, organizational role, etc.)
  - **Example**: Researchers who focus on the same research area are more likely to establish a connection (meeting at conferences, interacting in academic talks, etc.)

## Homophily

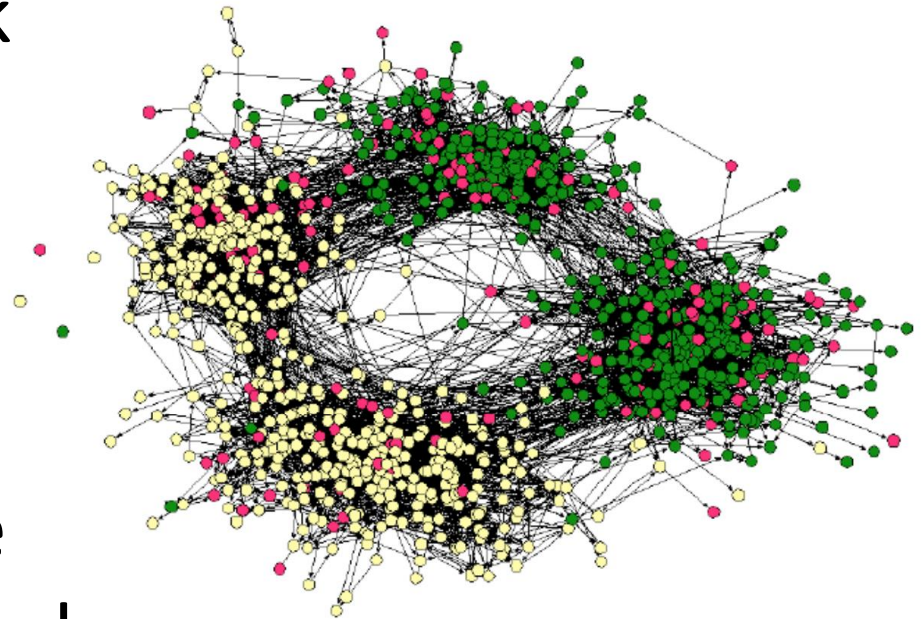




# Homophily: Example

## Example of homophily

- Online social network
  - Nodes = people
  - Edges = friendship
  - Node color = interests (sports, arts, etc.)
- People with the same interest are more closely connected due to homophily

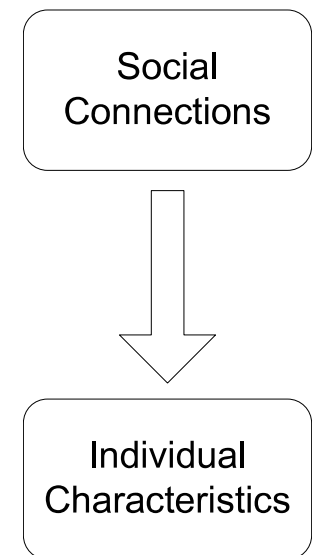


(Easley and Kleinberg, 2010)

# Social Influence: Example

- **Influence:** Social connections can influence the individual characteristics of a person.
- **Example:** I recommend my musical preferences to my friends, until one of them grows to like my same favorite genres!

## Influence



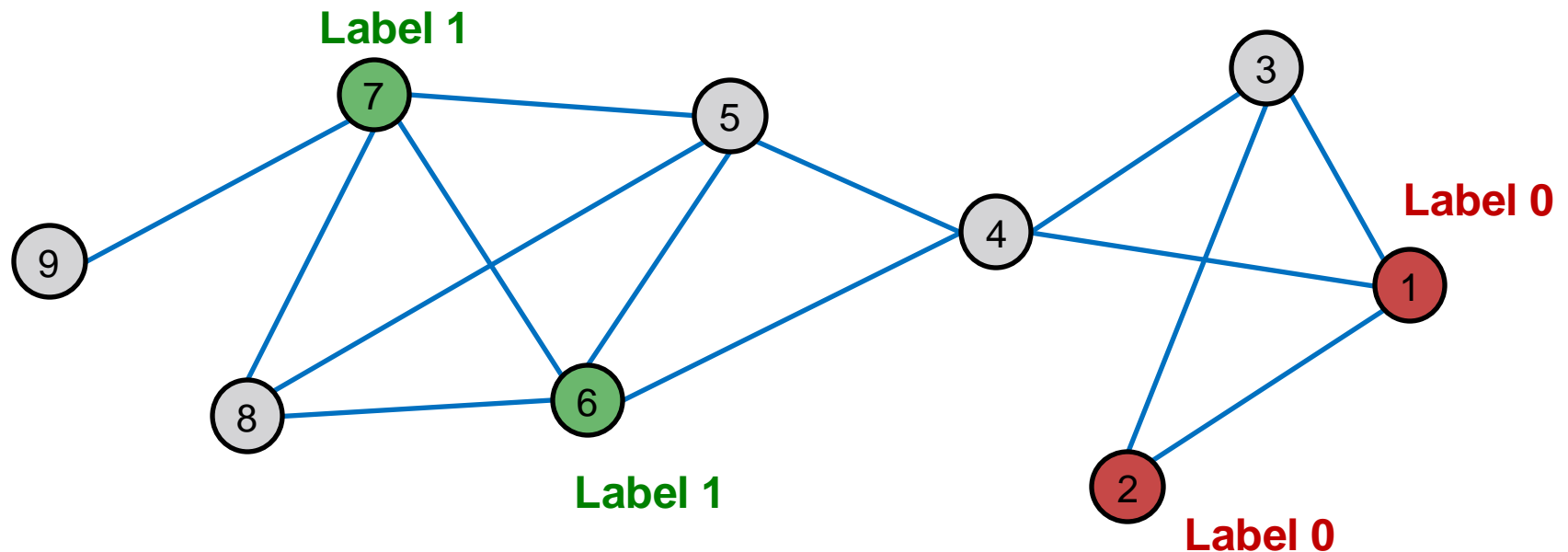
# Stanford CS224W: How do we leverage node correlations in networks?

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



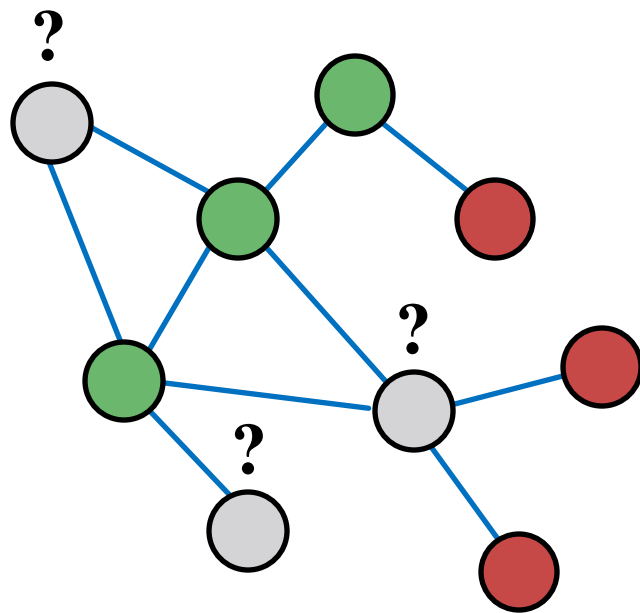
# Classification with Network Data

- How do we **leverage this correlation** observed in networks to help predict node labels?



How do we predict the labels for the nodes in grey?

# Semi-supervised Learning (1)



**Formal setting:**

**Given:**

- Graph
- Few labeled nodes

**Find:** Class (red/green) of remaining nodes

**Main assumption:** There is homophily in the network

# Semi-supervised Learning (2)

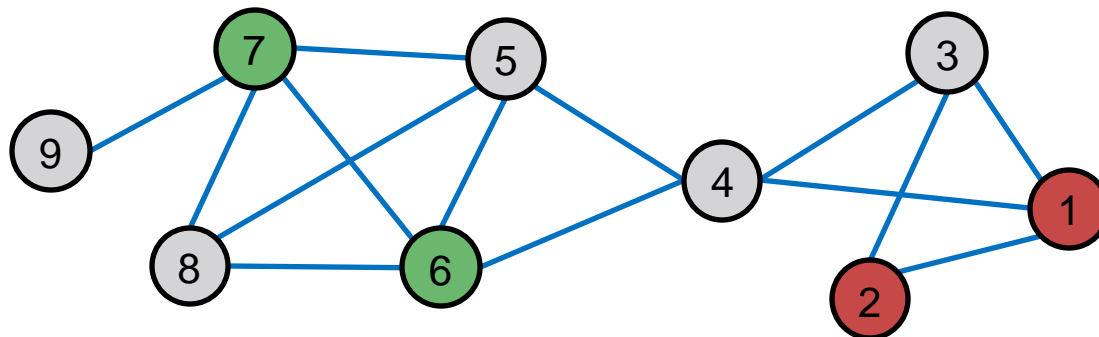
## Example task:

- Let  $A$  be a  $n \times n$  adjacency matrix over  $n$  nodes
- Let  $Y = \{0, 1\}^n$  be a vector of **labels**:
  - $Y_v = 1$  belongs to **Class 1**
  - $Y_v = 0$  belongs to **Class 0**
  - There are **unlabeled** node needs to be classified
- **Goal:** Predict which **unlabeled** nodes are likely **Class 1**, and which are likely **Class 0**

# Problem Setting

- How to predict the labels  $Y_v$  for the unlabeled nodes  $v$  (in grey color)?
- Each node  $v$  has a feature vector  $f_v$
- Labels for some nodes are given (1 for green, 0 for red)
- **Task:** Find  $P(Y_v)$  given all features and the network

$$P(Y_v) = ?$$



# Overview of What is Coming

- We focus on **semi-supervised binary node classification**
- We will introduce three approaches:
  - **Label propagation**
  - **Correct & Smooth**
  - **Masked label prediction**



# Stanford CS224W: Label Propagation

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# Label Propagation (1)

- **Idea:** Propagate node labels across the network
  - Class probability  $Y_v$  of node  $v$  is a weighted average of class probabilities of its neighbors.
- For **labeled nodes**  $v$ , initialize label  $Y_v$  with ground-truth label  $Y_v^*$ .
- For **unlabeled nodes**, initialize  $Y_v = 0.5$ .
- **Update all unlabeled nodes** (in parallel or a random order) until convergence or until maximum number of iterations is reached.

# Label Propagation (2)

- **The math:** Update each unlabeled node  $v$  label

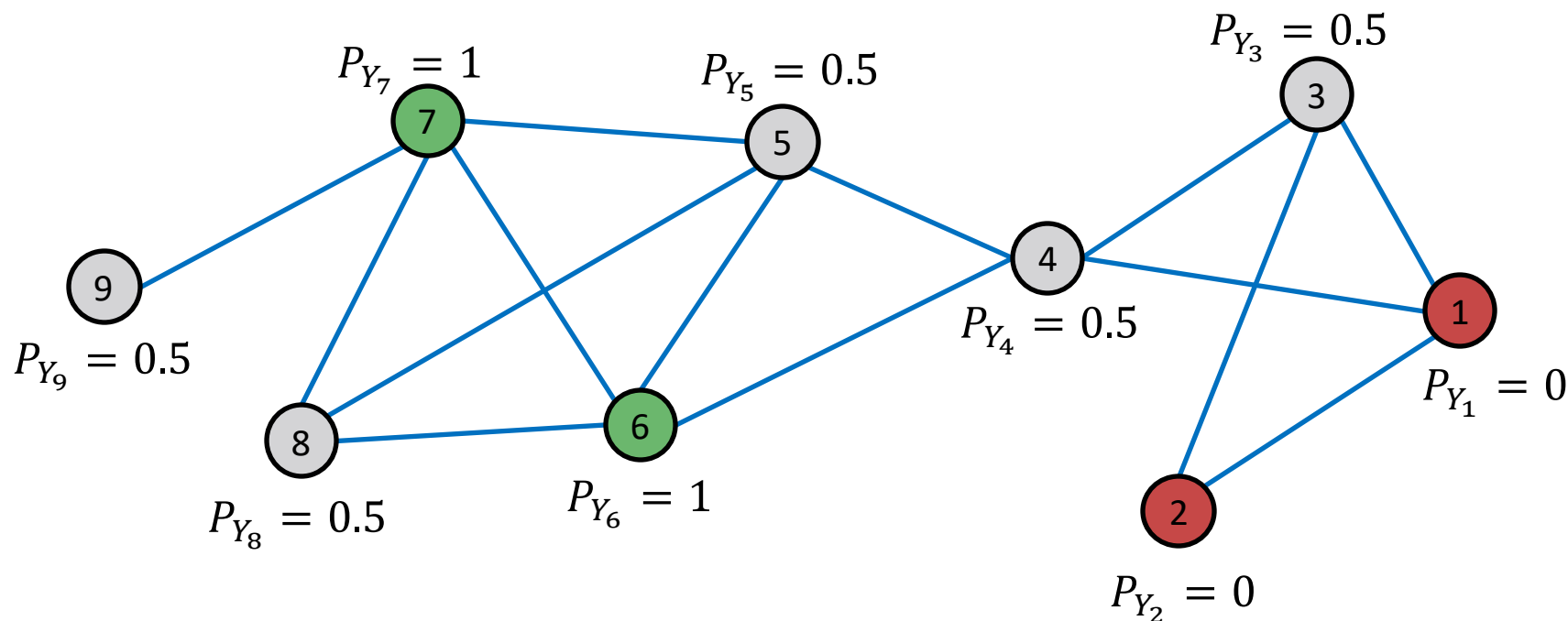
$$P^{(t+1)}(Y_v = c) = \frac{1}{\sum_{(v,u) \in E} A_{v,u}} \sum_{(v,u) \in E} A_{v,u} P^{(t)}(Y_u = c)$$

- If edges have strength/weight information,  $A_{v,u}$  can be the edge weight between  $v$  and  $u$ .
- $P(Y_v = c)$  is the probability of node  $v$  having label  $c$ .
- **Repeated** the update until convergence

# Example: Initialization

## Initialization:

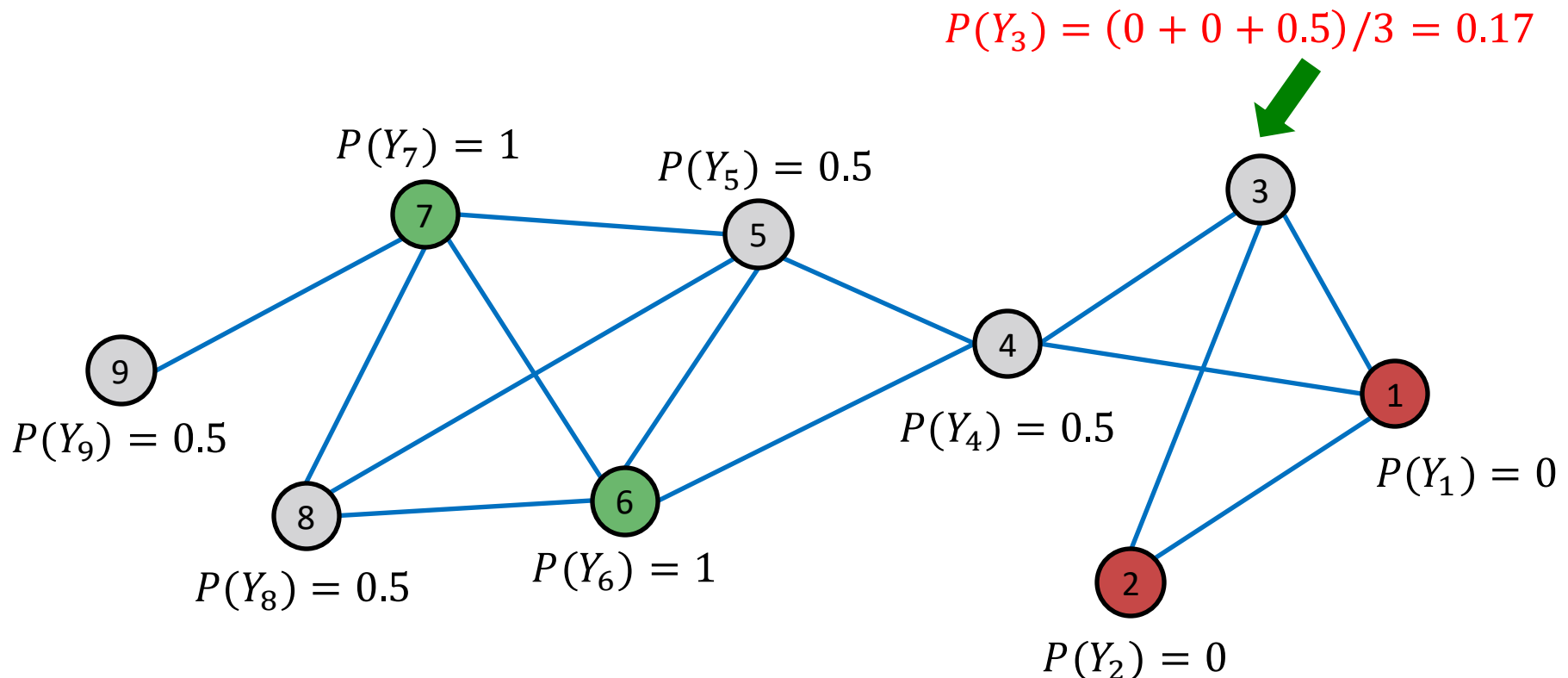
- All labeled nodes with their labels
- All unlabeled nodes 0.5 (belonging to class 1 with probability 0.5)



Notation:  $P_{Y_1} = P(Y_1 = 1)$

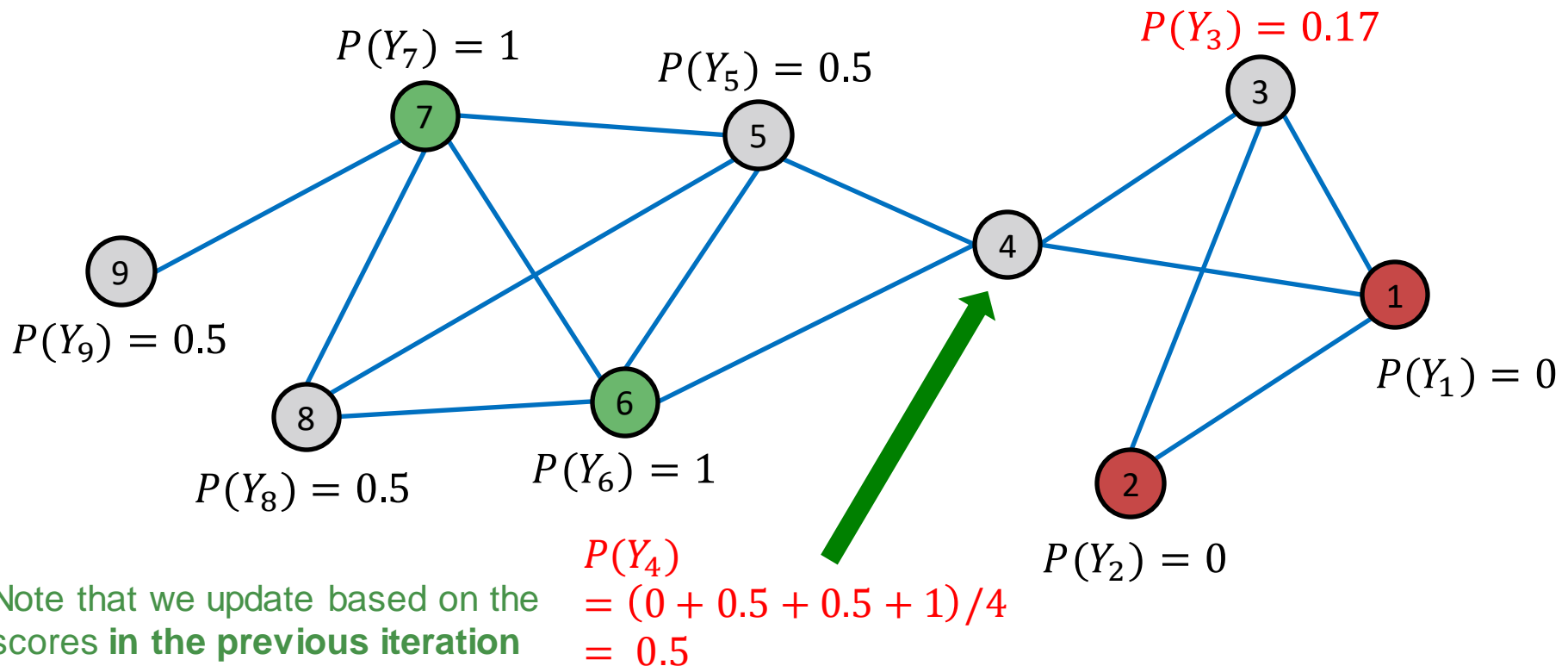
# Example: 1<sup>st</sup> Iteration, Update Node 3

- Update for the 1<sup>st</sup> Iteration:
  - For node 3,  $N_3 = \{1, 2, 4\}$



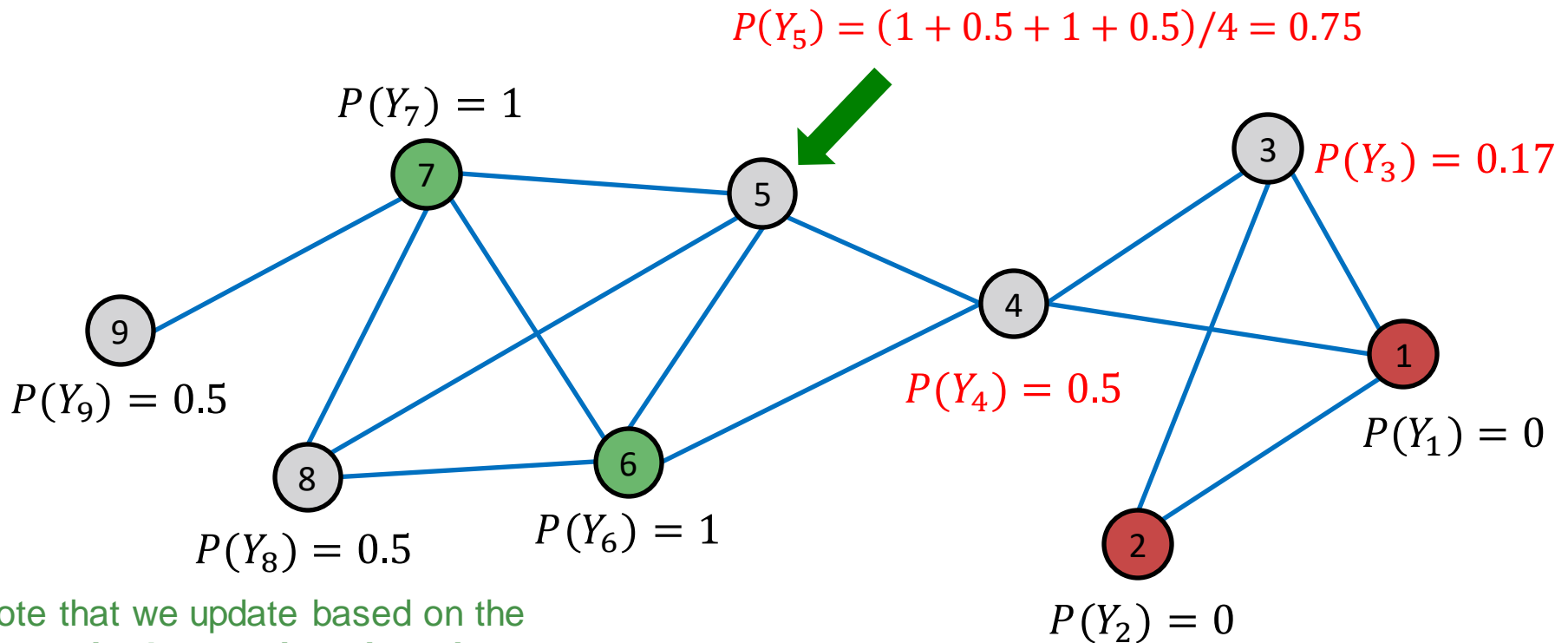
# Example: 1<sup>st</sup> Iteration, Update Node 4

- Update for the 1<sup>st</sup> Iteration:
  - For node 4,  $N_4 = \{1, 3, 5, 6\}$



# Example: 1<sup>st</sup> Iteration, Update Node 5

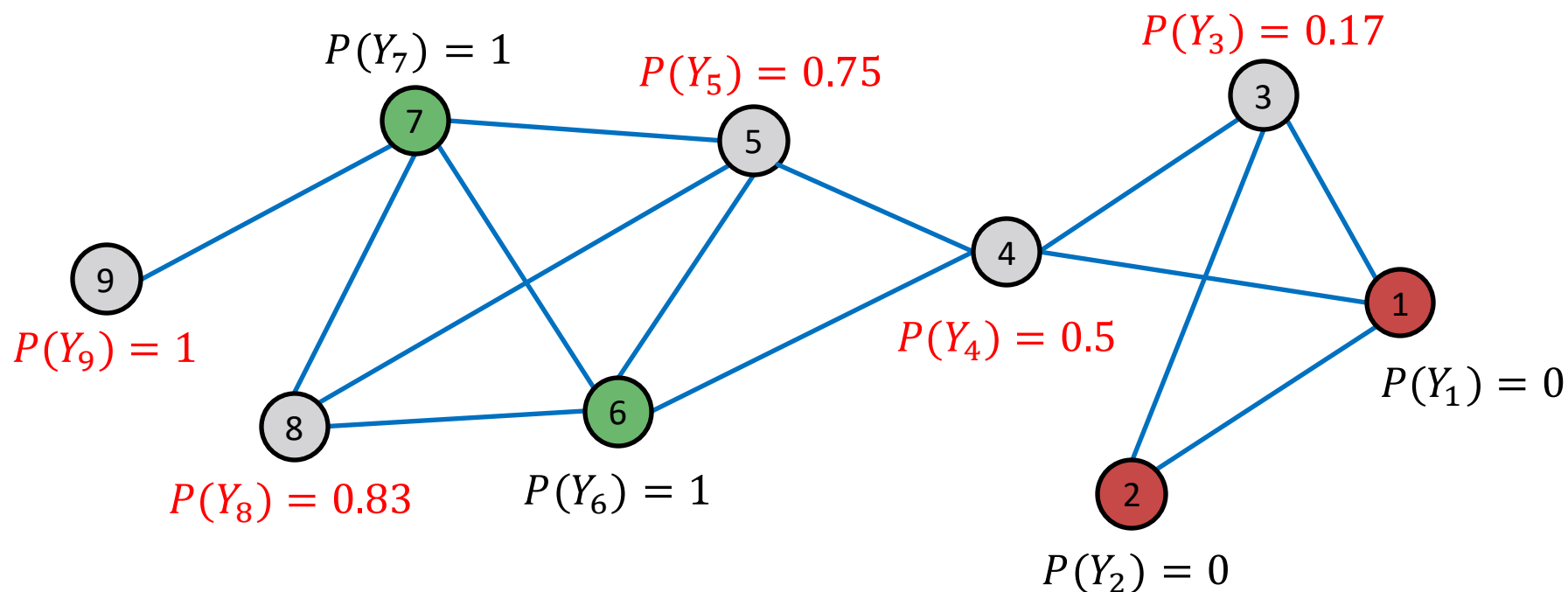
- Update for the 1<sup>st</sup> Iteration:
  - For node 5,  $N_5 = \{4, 6, 7, 8\}$



Note that we update based on the scores in the previous iteration

# Example: After 1<sup>st</sup> Iteration

After Iteration 1 (a round of updates for all unlabeled nodes)

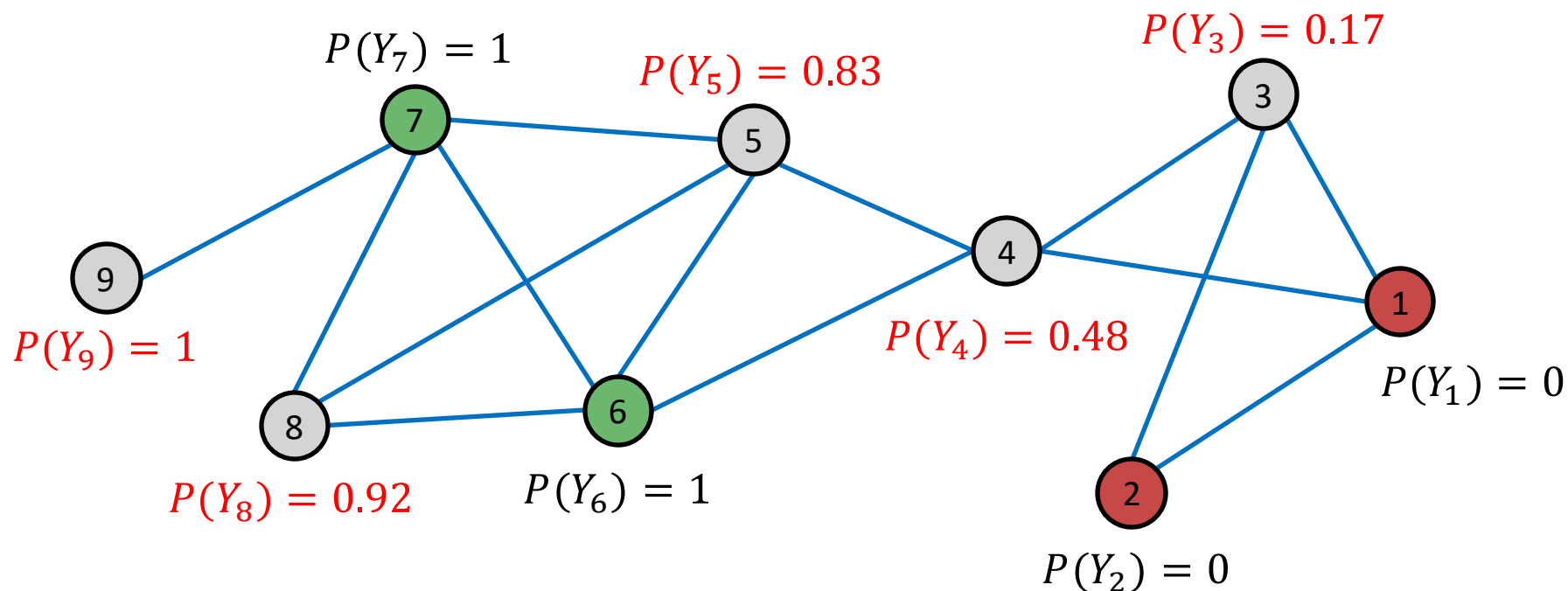


Note that we update based on the scores in the previous iteration



# Example: After 2<sup>nd</sup> Iteration

After Iteration 2



# Label Propagation: Convergence

- **Intuition:** Stop iterating when all nodes have reached a static value. Applying label propagation does not change labels anymore.
- **Convergence criteria:**

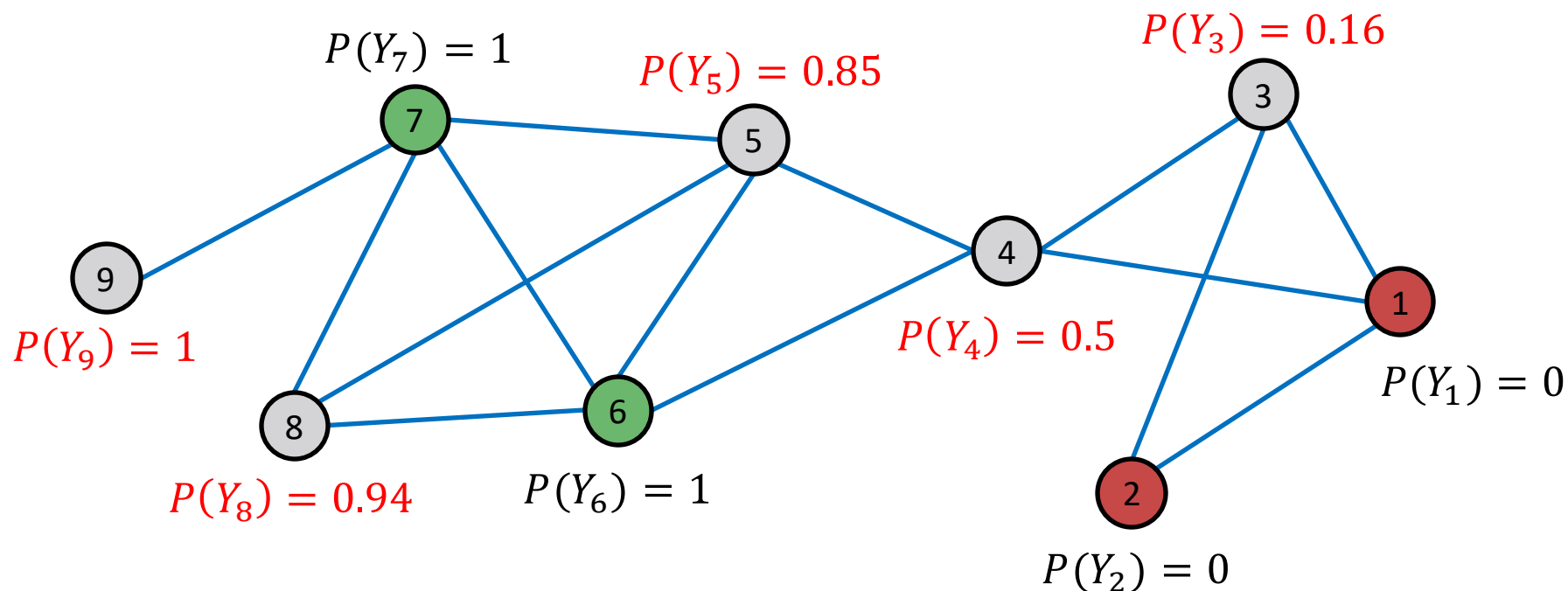
$$\left| P^{(t)}(Y_v) - P^{(t-1)}(Y_v) \right| \leq \epsilon$$

for all nodes  $v$ .

Here  $\epsilon$  is the convergence threshold.

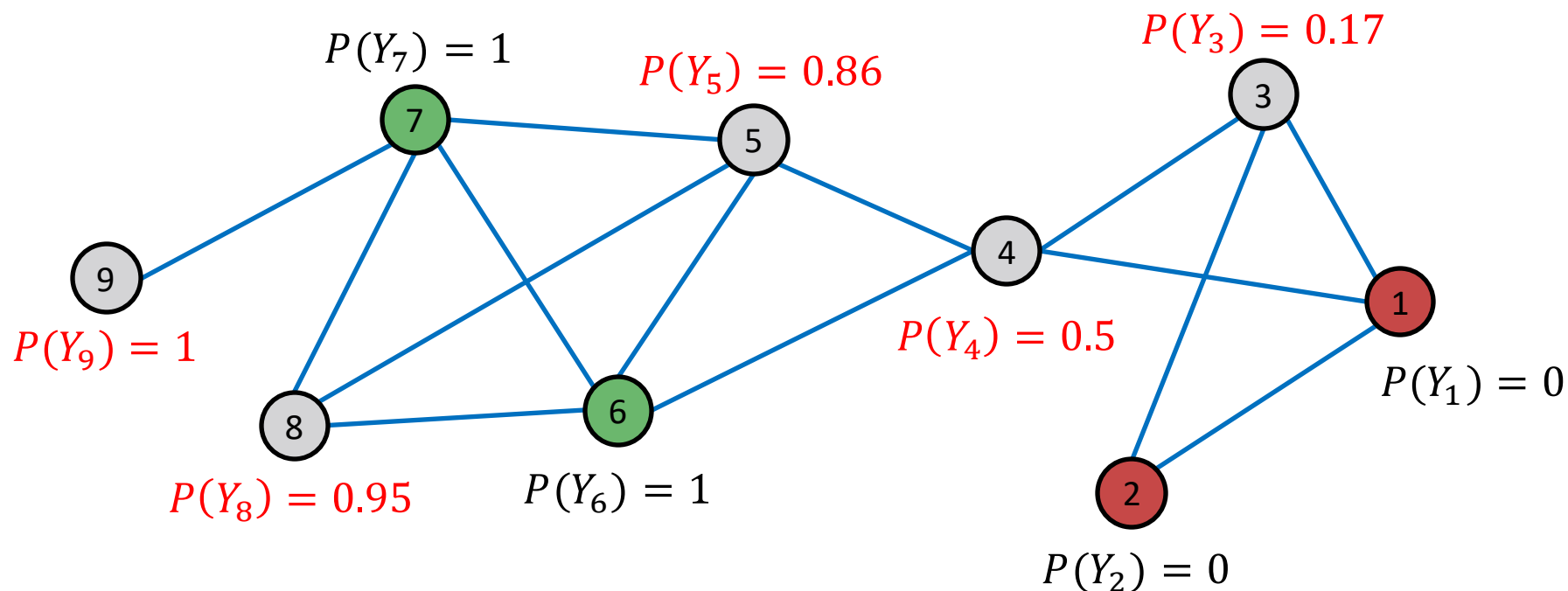
# Example: After 3<sup>rd</sup> Iteration

After Iteration 3



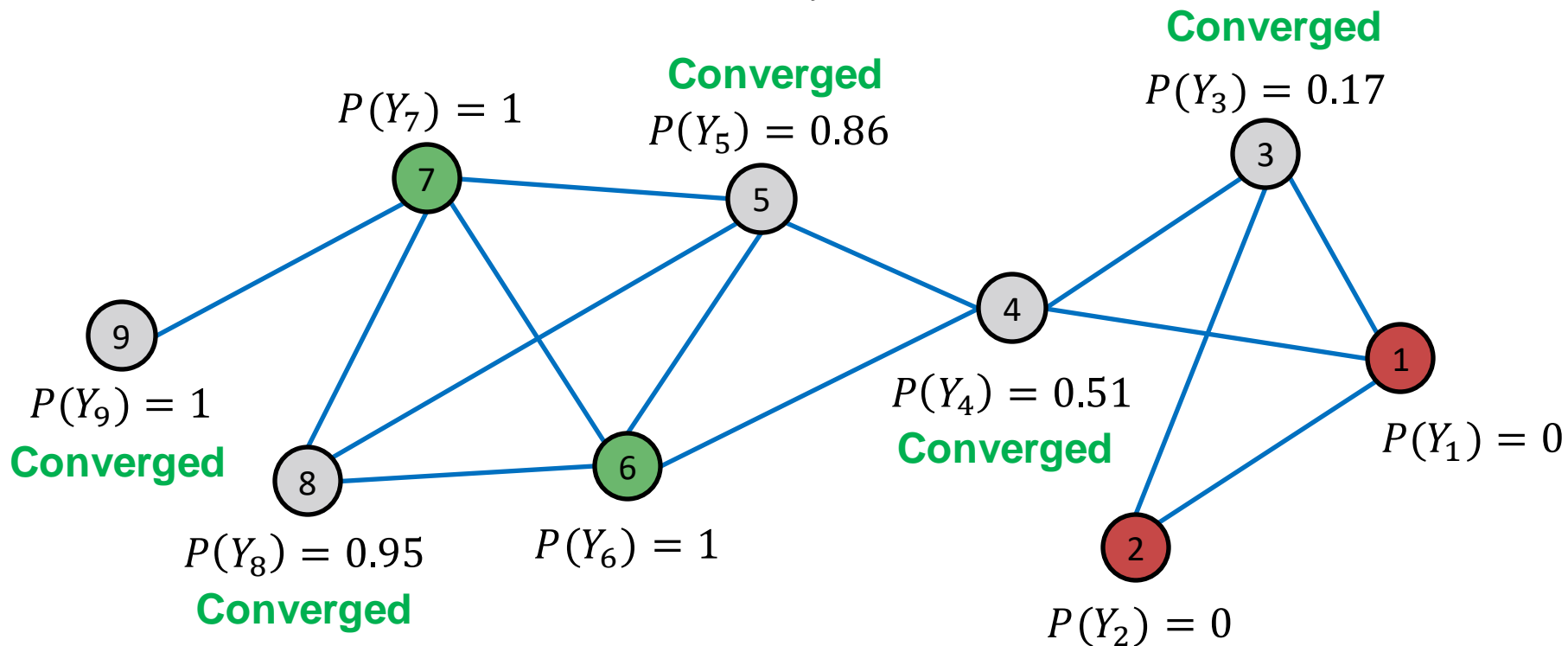
# Example: After 4<sup>th</sup> Iteration

After Iteration 4



# Example: Convergence

- All scores stabilize after 4 iterations. We therefore predict:
  - **Nodes 4, 5, 8, 9 belong to class 1** ( $P_{Y_v} > 0.5$ )
  - **Nodes 3 belong to class 0** ( $P_{Y_v} < 0.5$ )



# Label Propagation Applications

- **Document classification**
  - Nodes: documents
  - Edges: document similarity (e.g. word overlap)
- **Twitter polarity classification**
  - Nodes: users, tweets, words, hashtags, emoji
  - Edges [users -> users] Twitter follower graph;  
Edges [users -> tweets] creation;  
Edges [tweets -> words/hashtags/emoji] part of.
- **Spam and fraud detection**

# Summary

- **Label propagation**

- Iteratively update probabilities of node belonging to a label class based on its neighbors

- **Issue**

- Convergence may be **very slow** and not **guaranteed**
- Label propagation **does not use node attributes**
- Can we improve the idea of label propagation to **leverage attribute/feature information?**

# Stanford CS224W: Node Classification: Correct & Smooth

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>





# Correct & Smooth

- **Correct & Smooth (C&S)**, recent state-of-the-art node classification method.

Leaderboard for [ogbn-products](#)

The classification accuracy on the test and validation sets. The higher, the better.

Package:  $\geq 1.1.1$

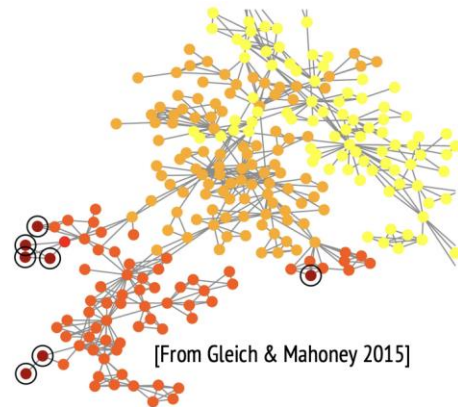
Rank	Method	Ext. data	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	GLEM+EnGCN	Yes	0.9014 $\pm$ 0.0012	0.9370 $\pm$ 0.0004	Jianan Zhao (Mila & MSRA Team)	<a href="#">Paper</a> , <a href="#">Code</a>	139,633,805	Tesla V100 (32GB)	Oct 27, 2022
2	EnGCN	No	0.8798 $\pm$ 0.0004	0.9241 $\pm$ 0.0003	Keyu Duan (NUS)	<a href="#">Paper</a> , <a href="#">Code</a>	653,918	GeForce RTX 3090 (24GB)	Oct 23, 2022
3	GLEM+GIANT+SAGN+SCR	Yes	0.8737 $\pm$ 0.0006	0.9400 $\pm$ 0.0003	Jianan Zhao (Mila & MSRA Team)	<a href="#">Paper</a> , <a href="#">Code</a>	139,792,525	Tesla V100 (32GB)	Oct 27, 2022
4	GIANT-XRT+R-SAGN+SCR+C&S	Yes	0.8684 $\pm$ 0.0005	0.9365 $\pm$ 0.0003	LeeXue (HIT Team)	<a href="#">Paper</a> , <a href="#">Code</a>	1,154,142	TITAN RTX (24GB GPU)	Sep 30, 2022

**C&S** tops the OGB leaderboard!

[OGB leaderboard](#) snapshot at Jan 30, 2023

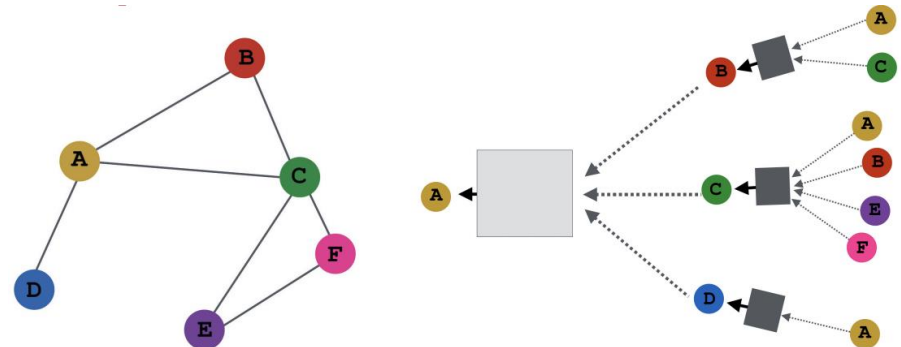
# Correct & Smooth: Motivations

## Label Propagation (LP)



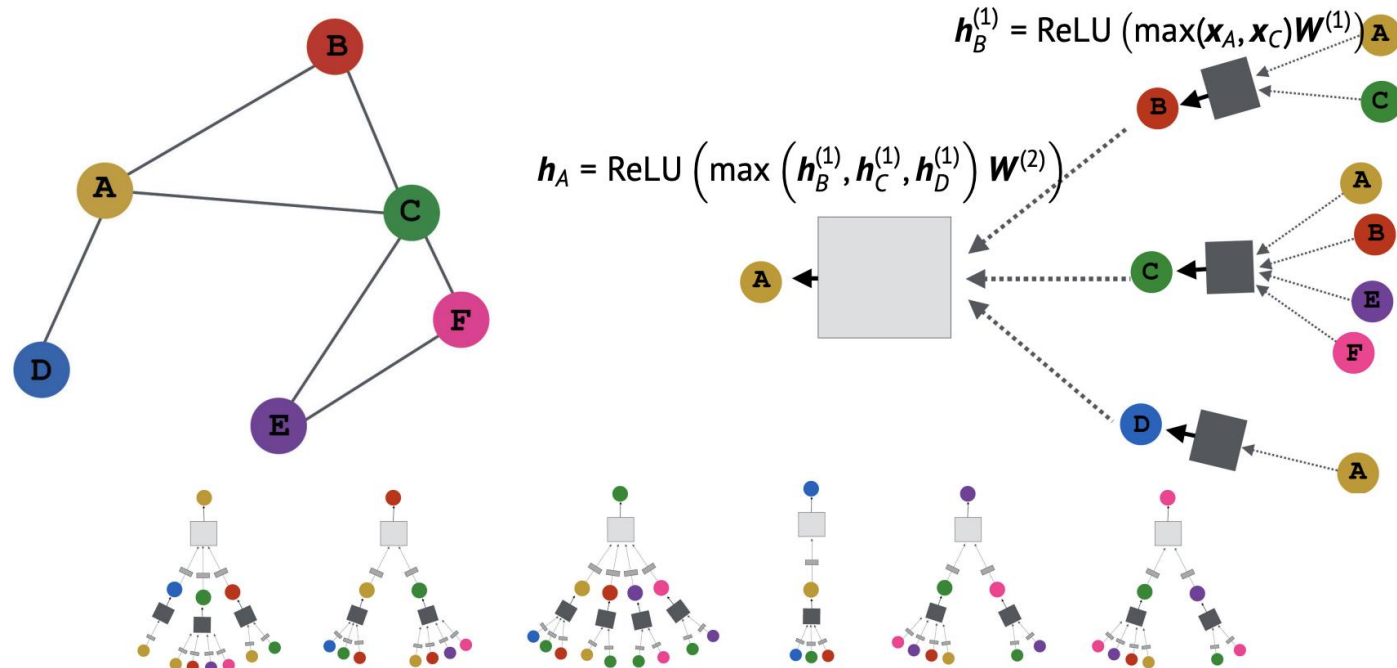
- Modeling assumption: connected nodes have similar labels.
- Works because of homophily a.k.a. associativity
- **FAST**: few sparse matrix-vector products
- **Why not use additional info or features?**

## Graph Neural Networks



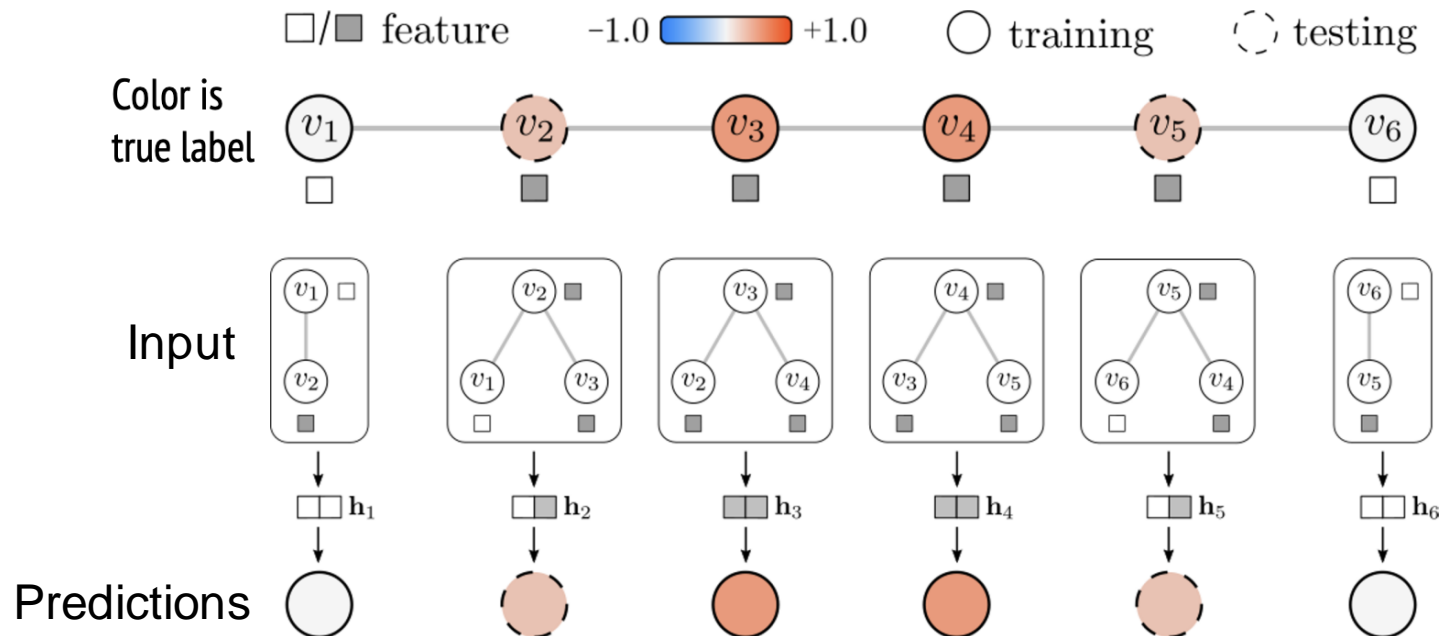
- Modeling assumption: labels only depend on neighbor features.
- Works because these features are sometimes very informative.
- **SLOW**: many parameters, irregular computation
- **Why not assume labels are correlated?**

# GNNs make uncorrelated predictions



- **GNN uses labels to train the weights** of a model
  - Given a trained model, **predictions for different node are independent/uncorrelated.**
- In contrast, **LP directly uses the labels in predictions.**

# GNNs make uncorrelated predictions

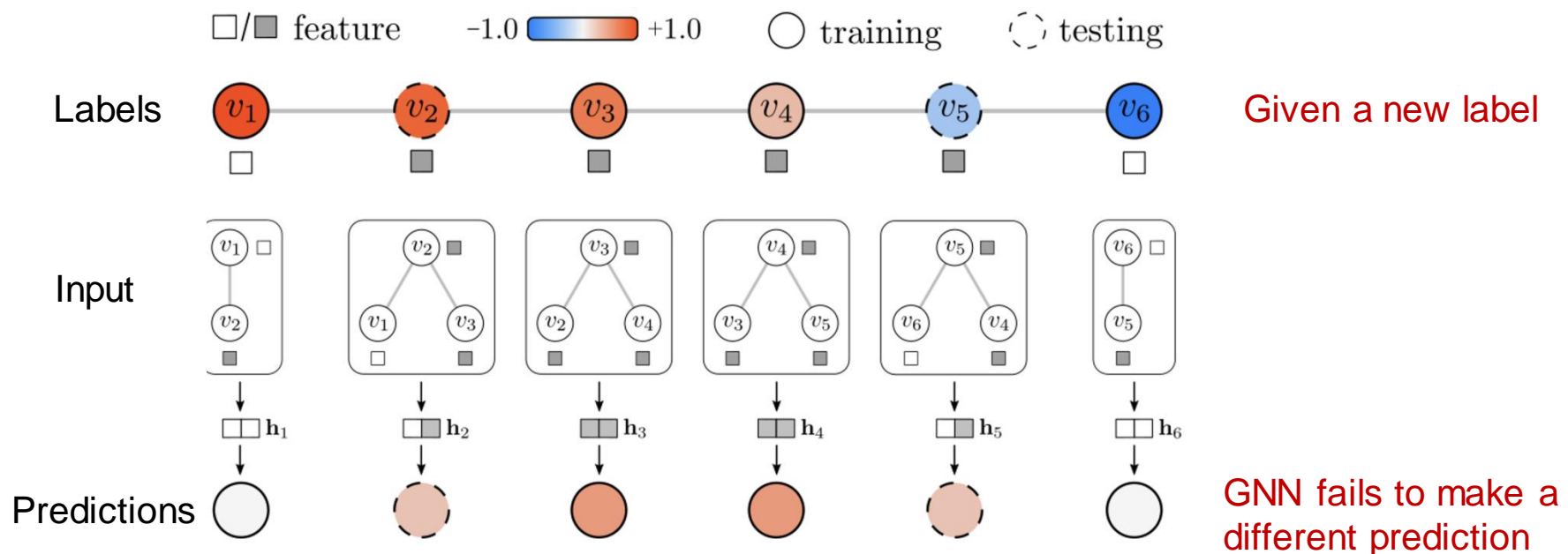


1. Form local neighborhoods.
2. Combine features to get a representation  $h_v$  at node  $v$ .
3. Predict outcome given representation (learn params with train data).

**If node features are overwhelmingly predictive, making uncorrelated predictions for each node is OK.**

# GNNs make uncorrelated predictions

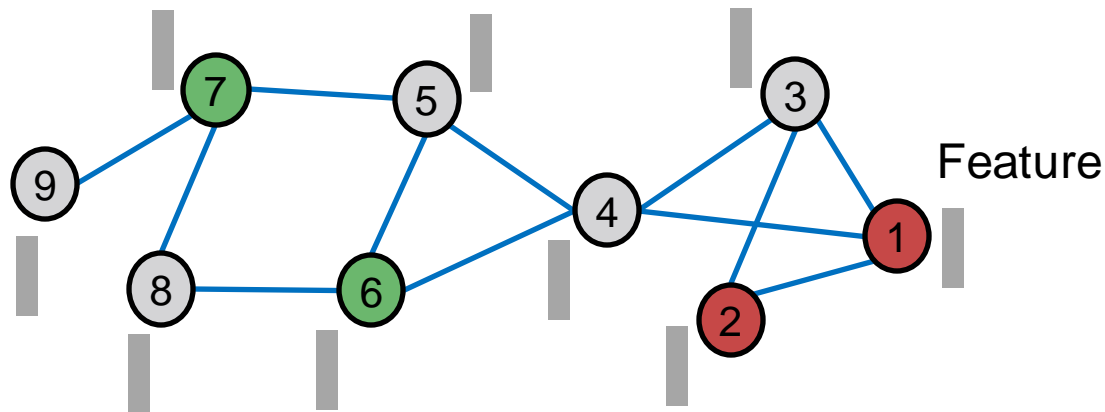
**Uncorrelated GNN predictions can be catastrophic** in simple cases **when features are only mildly predictive.**



- **Big problem!** Features are no longer super predictive.
- LP (ignoring features) would work much better.

# Correct & Smooth

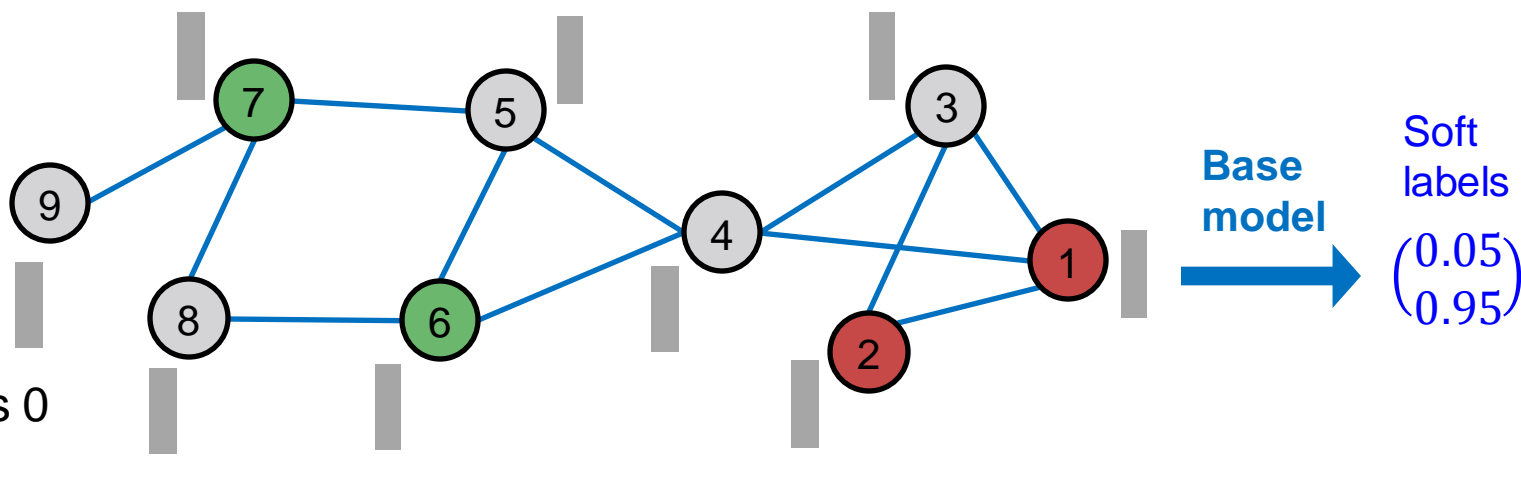
- **Setting:** A partially labeled graph and features over nodes.



- **C&S follows the three-step procedure:**
  1. Train base predictor
  2. Use the base predictor to **predict soft labels** of all nodes.
  3. **Post-process the predictions using graph structure** to obtain the final predictions of all nodes.

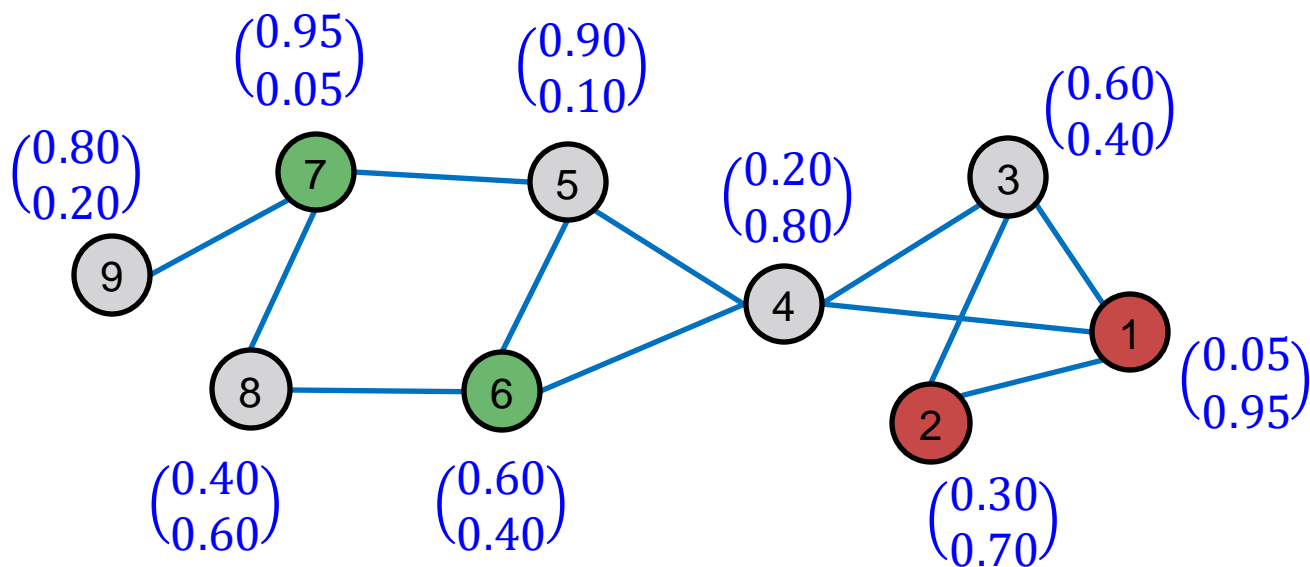
# C&S: (1) Train Base Predictor

- (1) Train a **base predictor** that predict **soft labels** (class probabilities) over all nodes.
  - Labeled nodes are used for train/validation data.
  - **Base predictor** can be simple:
    - Linear model/Multi-Layer-Perceptron(MLP) over node features, or a full GNN



# C&S: (2) Predict Over All Nodes

- (2) Given a trained **base predictor**, we apply it to obtain **soft labels** for all the nodes.
  - We expect these soft labels to be decently accurate.
  - Can we use graph structure to post-process the predictions to make them more accurate?

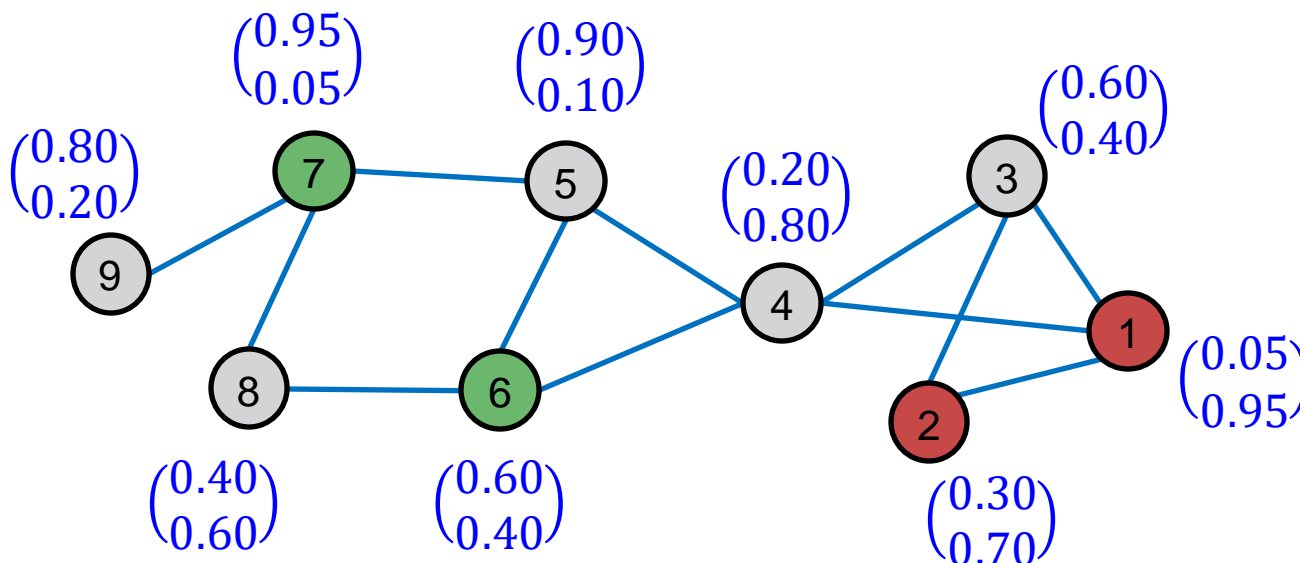




# C&S: (3) Post-Process Predictions

- (3) C&S uses the 2-step procedure to post-process the **soft predictions**.

1. **Correct step**
2. **Smooth step**



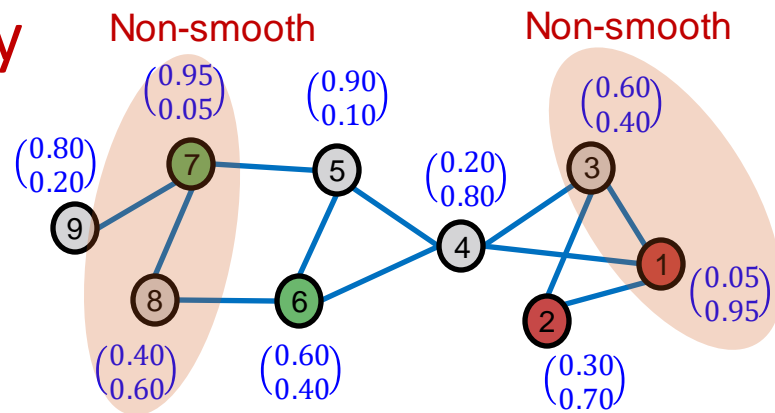
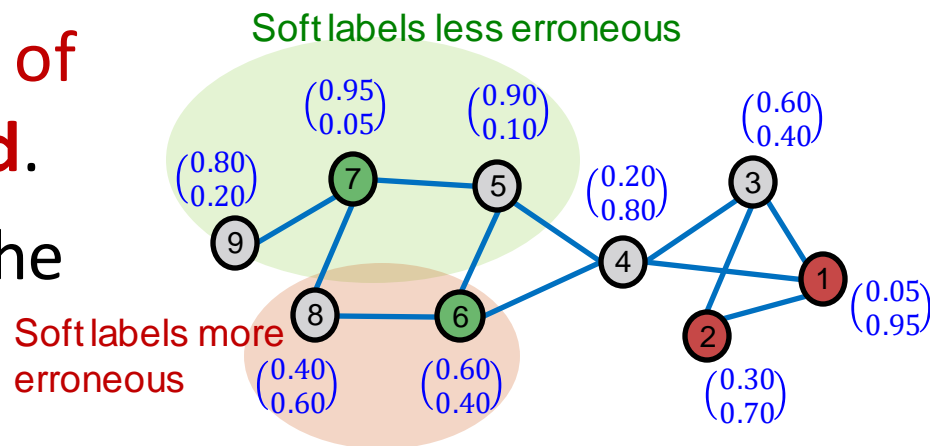
# Intuition of Correct & Smooth

## ■ Correct step

- The degree of the **errors of the soft labels are biased**.
- We need to correct for the error bias.

## ■ Smooth step

- The predicted **soft labels may not be smooth** over the graph.
- We need to smoothen the soft labels.



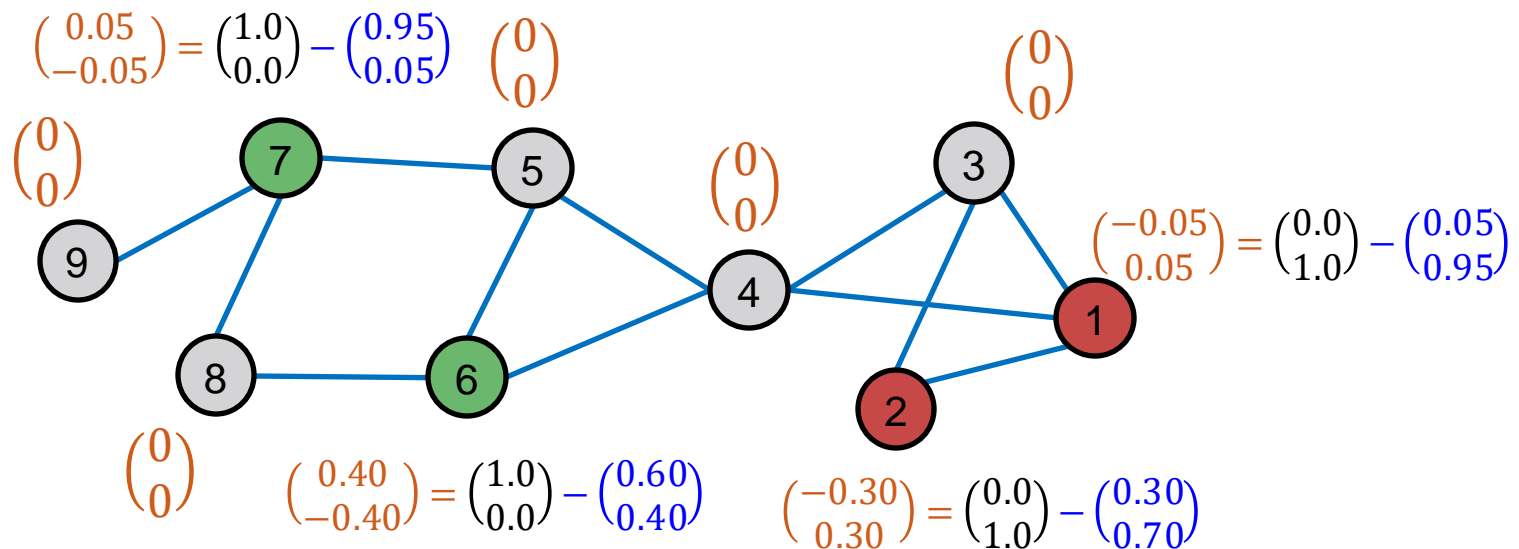
# C&S Post-Processing: Correct Step

- The key idea is that we expect **errors** in the base prediction to be **positively correlated along edges in the graph**.
  - In other words, an error at node  $u$  increases the chance of a similar error at neighbors of  $u$ .
  - Thus, we should “spread” such uncertainty over the graph.

# C&S Post-Processing: Correct Step (1)

## ■ Correct step:

- Compute **training errors** of nodes.
  - **Training error**: Ground-truth label minus soft label. Defined as 0 for unlabeled nodes.



# C&S Post-Processing: Correct Step (2)

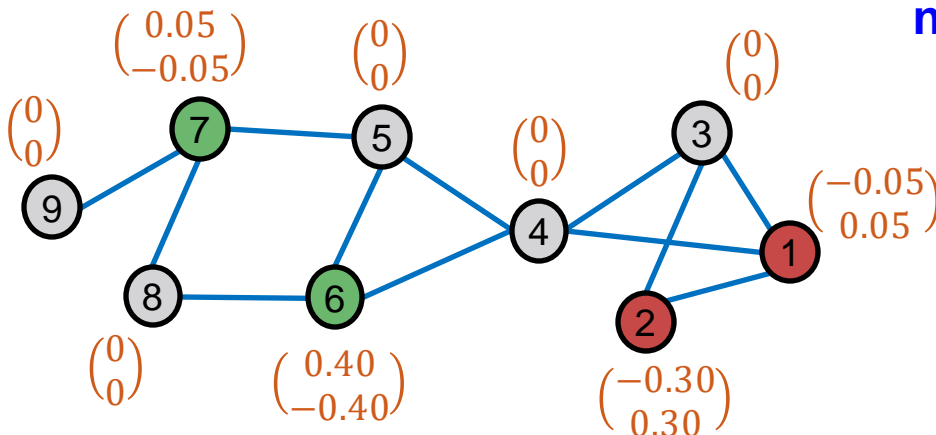
## ■ Correct step (contd.):

- Diffuse **training errors**  $E^{(0)}$  along the edges.
- Let  $A$  be the adjacency matrix,  $\tilde{A}$  be the **diffusion matrix** (defined in the next slide).

$$\text{■ } E^{(t+1)} \leftarrow (1 - \overset{\text{Hyper-parameter}}{\alpha}) \cdot E^{(t)} + \alpha \cdot \tilde{A} E^{(t)}.$$

- Similar to PageRank.

Diffuse training errors along the edges  
**Assumption: errors are similar for nearby nodes**

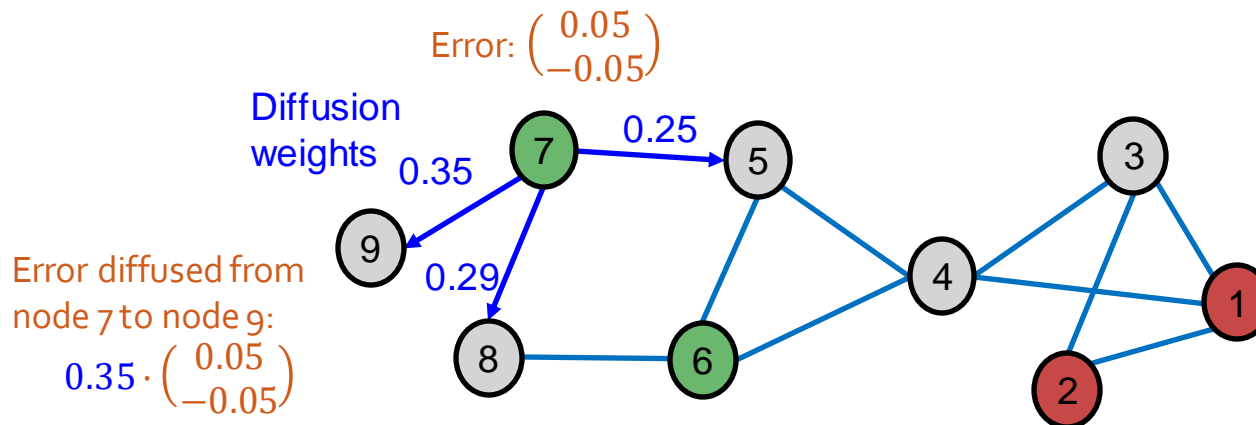


Initial  
training  
error  
matrix

$$E^{(0)} = \begin{pmatrix} -0.05 & 0.05 \\ -0.30 & 0.30 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.40 & -0.40 \\ 0.05 & -0.05 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

# Diffusion Matrix $\tilde{A}$

- **Normalized diffusion matrix  $\tilde{A} \equiv \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$** 
  - Add self-loop to the adjacency matrix  $\mathbf{A}$ , i.e.,  $A_{ii} = 1$ .
  - Let  $\mathbf{D} \equiv \text{Diag}(d_1, \dots, d_N)$  be the degree matrix.
  - See [Zhu et al. ICML 2013](#) for details.

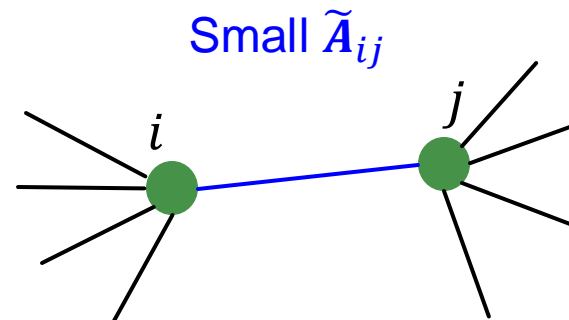
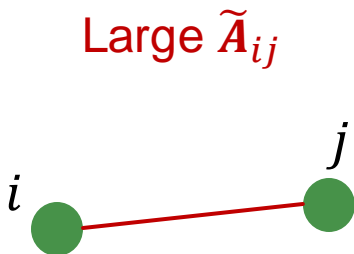


# Theoretical Motivation for $\tilde{A}$

- **Normalized diffusion matrix**  $\tilde{A} \equiv D^{-1/2} A D^{-1/2}$
- All the eigenvalues  $\lambda$ 's are in the range of  $[-1,1]$ .
  - Eigenvector for  $\lambda = 1$  is  $D^{1/2} \mathbf{1}$  ( $\mathbf{1}$  is an all-one vector).
    - Proof:  $\tilde{A} D^{1/2} \mathbf{1} = D^{-1/2} A D^{-1/2} D^{1/2} \mathbf{1} = D^{-1/2} A \mathbf{1} = D^{-1/2} D \mathbf{1} = \mathbf{1} \cdot D^{1/2} \mathbf{1}$ .
      - $A \mathbf{1} = D \mathbf{1}$ , since they both leads to node degree vector
  - The power of  $\tilde{A}$  (i.e.,  $\tilde{A}^K$ ) is **well-behaved for any  $K$** .
    - The eigenvalues of  $\tilde{A}^K$  are always within  $[-1,1]$ .
    - The largest eigenvalue is always 1.
    - Output of  $\tilde{A}x$  (e.g., in error diffusion) is **normalized**

# More Intuitions for $\tilde{A}$

- If  $i$  and  $j$  are connected, the weight  $\tilde{A}_{ij}$  is  $\frac{1}{\sqrt{d_i}\sqrt{d_j}}$
- **Intuition:**
  - **Large** if  $i$  and  $j$  are connected only with each other (no other nodes are connected to  $i$  and  $j$ ).
  - **Small** if  $i$  and  $j$  are connected also connected with many other nodes.





# C&S Post-Processing: Correct Step (3)

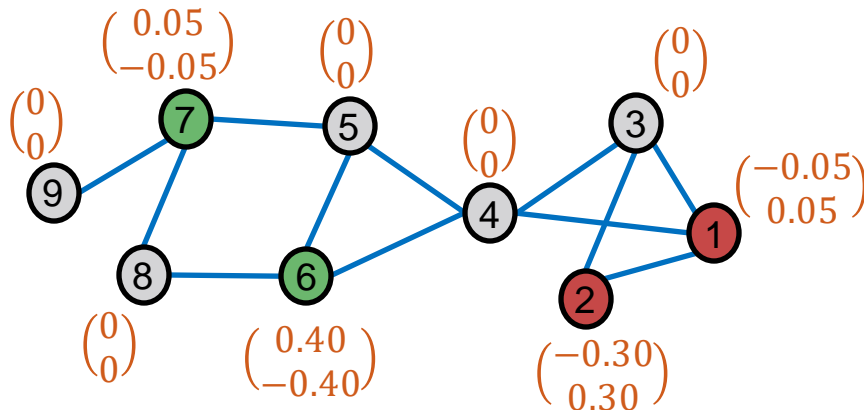
## ■ Diffusion of training errors:

$$E^{(t+1)} \leftarrow (1 - \alpha) \cdot E^{(t)} + \alpha \cdot \tilde{A}E^{(t)}$$

**Assumption:** Prediction errors are similar for nearby nodes.

**Before diffusion**

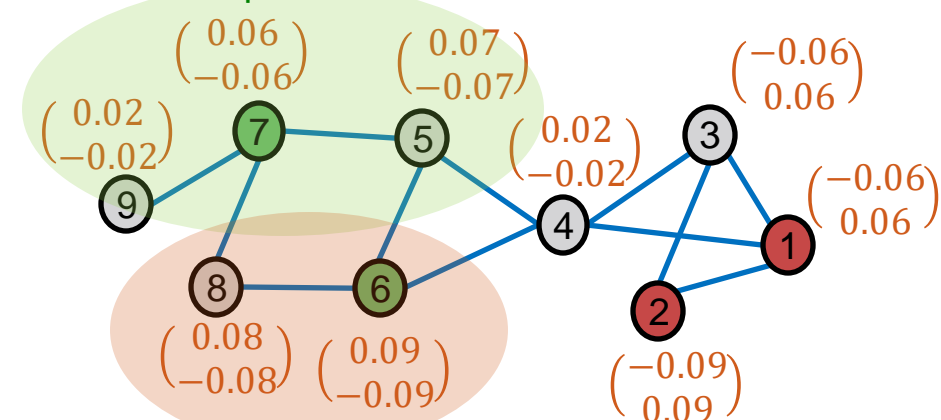
$E^{(0)}$



**After diffusion**

$E^{(3)}, \alpha = 0.8$

Less erroneous part

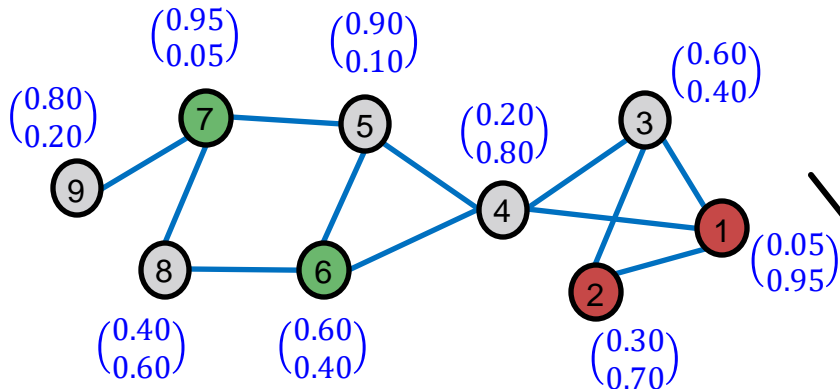


More erroneous part

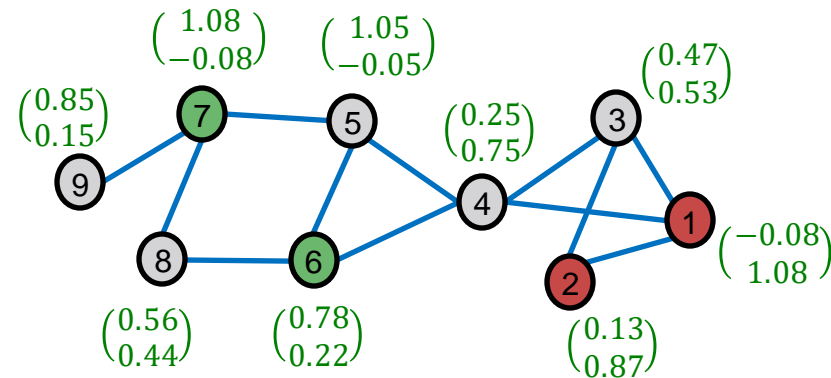
# C&S Post-Processing: Correct Step (4)

- Add the scaled **diffused training errors** into the predicted **soft labels**

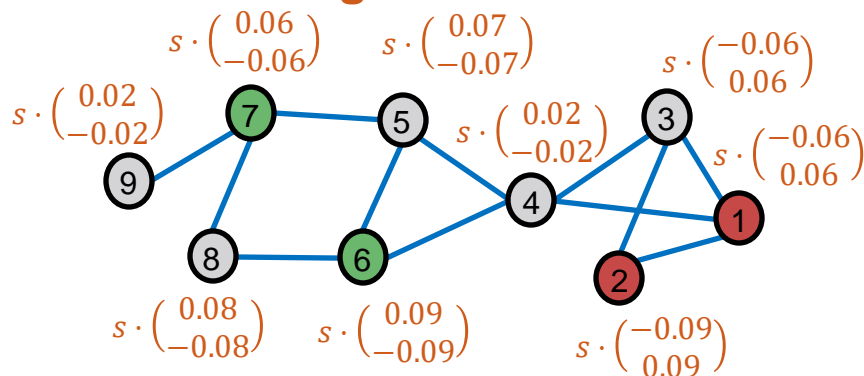
Soft labels



Output after the correct step ( $s = 2$ )



Diffused training errors



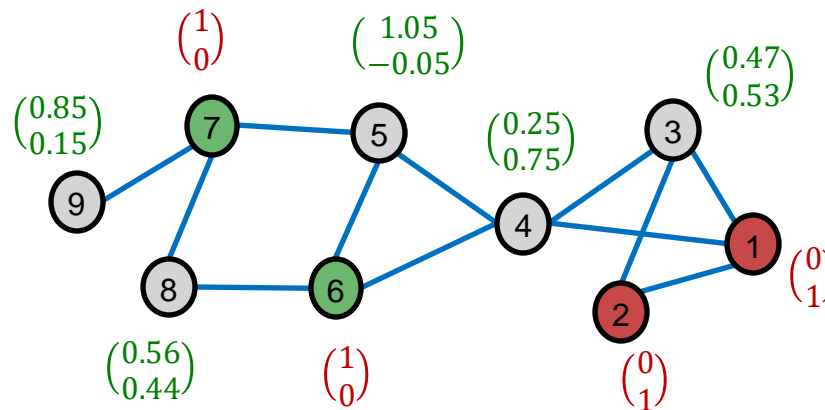
Scale by  $s$   
(hyper-parameter)

+

# C&S Post-Processing: Smooth Step

- Smoothen the corrected soft labels along the edges.
  - **Assumption**: Neighboring nodes tend to share the same labels.
  - **Note**: For training nodes, we use the **ground-truth hard labels** instead of the soft labels.

Input to the smooth step:



# C&S Post-Processing: Smooth Step (1)

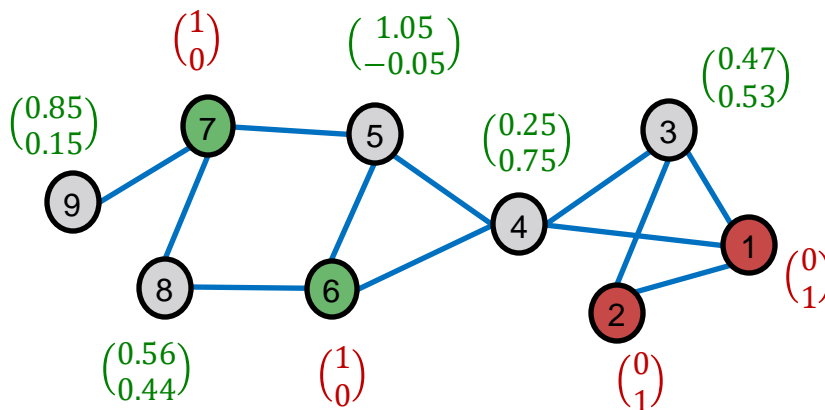
## ■ Smooth step:

- Diffuse label  $\mathbf{Z}^{(0)}$  along the graph structure.

- $\mathbf{Z}^{(t+1)} \leftarrow (1 - \alpha) \cdot \mathbf{Z}^{(t)} + \alpha \cdot \tilde{\mathbf{A}}\mathbf{Z}^{(t)}$ .

Hyper-parameter

Diffuse labels along the edges



Corrected  
label  
matrix

$$\mathbf{Z}^{(0)} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0.47 & 0.53 \\ 0.25 & 0.75 \\ 1.05 & -0.05 \\ 1 & 0 \\ 1 & 0 \\ 0.56 & 0.44 \\ 0.85 & 0.15 \end{pmatrix}$$

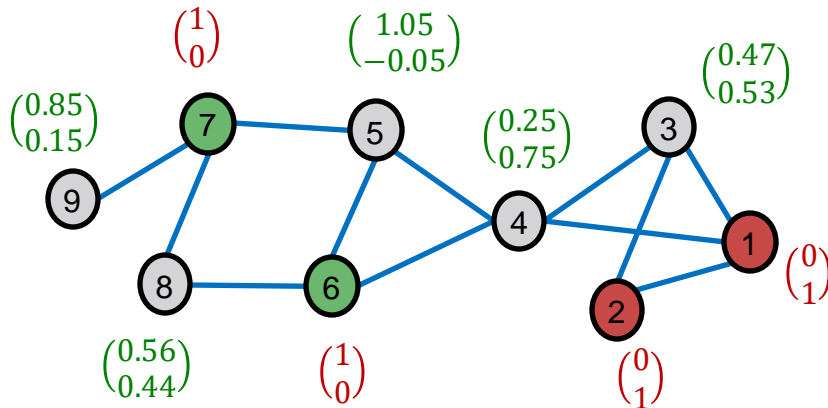
# C&S Post-Processing: Correct Step (2)

## Smooth step:

$$\mathbf{Z}^{(t+1)} \leftarrow (1 - \alpha) \cdot \mathbf{Z}^{(t)} + \alpha \cdot \tilde{\mathbf{A}}\mathbf{Z}^{(t)}$$

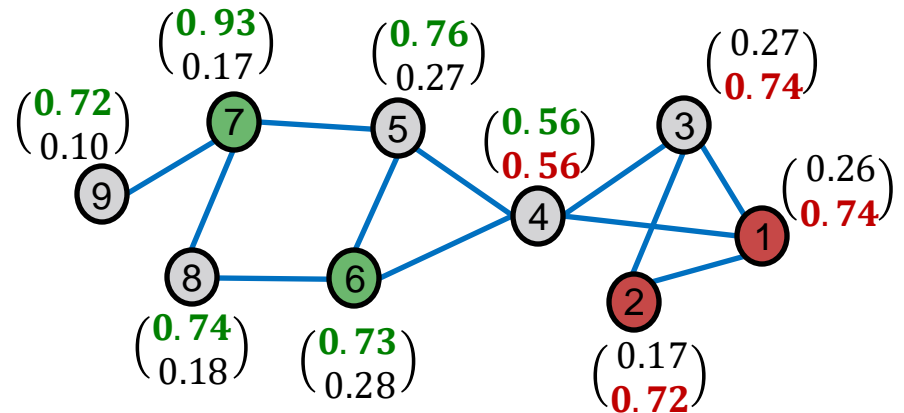
Before smoothing

$\mathbf{Z}^{(0)}$



After smoothing

$\mathbf{Z}^{(3)}, \alpha = 0.8$



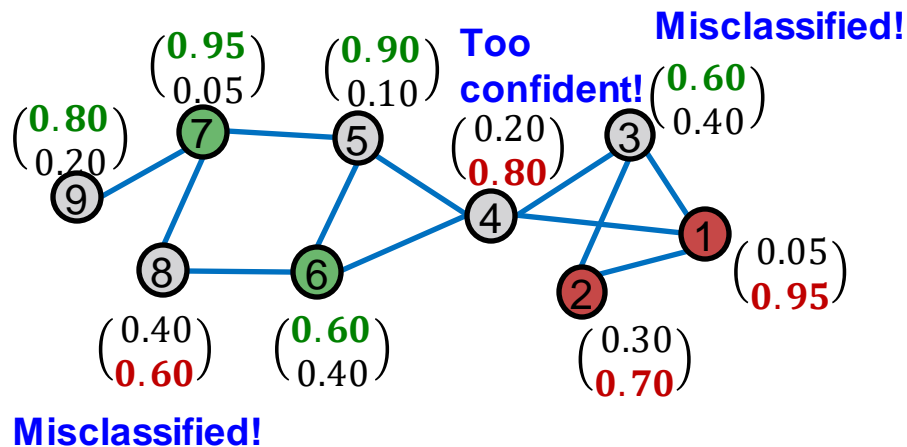
The final class prediction of C&S is the class with the maximum  $\mathbf{Z}^{(3)}$  score.

**Note:** The  $\mathbf{Z}^{(3)}$  scores do not have direct probabilistic interpretation (e.g., not sum to 1 for each node), but larger scores indicate the classes are more likely.

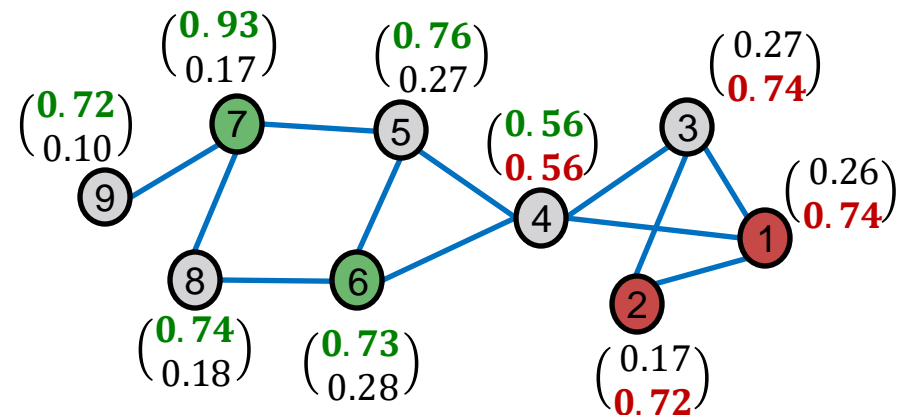
# C&S: Toy Example Summary

- Our toy example shows that C&S successfully improves base model performance using graph structure.

## Prediction of the base model



## After C&S



# C&S on a Real-World Dataset

- C&S significantly improves the performance of the base model (*e.g.*, MLP).
- C&S outperforms Smooth-only (no correct step) baseline.

Method	Classification accuracy (%) on ogbn-products dataset
MLP (base model)	63.41
MLP + smooth only	80.34
<b>MLP + C&amp;S</b>	<b>84.18</b>

# Correct & Smooth: Summary

- Correct & Smooth (C&S) **uses graph structure to post-process** the soft node labels predicted by any base model.
- **Correction step**: Diffuse and correct for the training errors of the base predictor.
- **Smooth step**: Smoothen the prediction of the base predictor (a variant of **label propagation**).
- C&S can be **combined with GNNs**
  - C&S achieves strong performance on semi-supervised node classification.



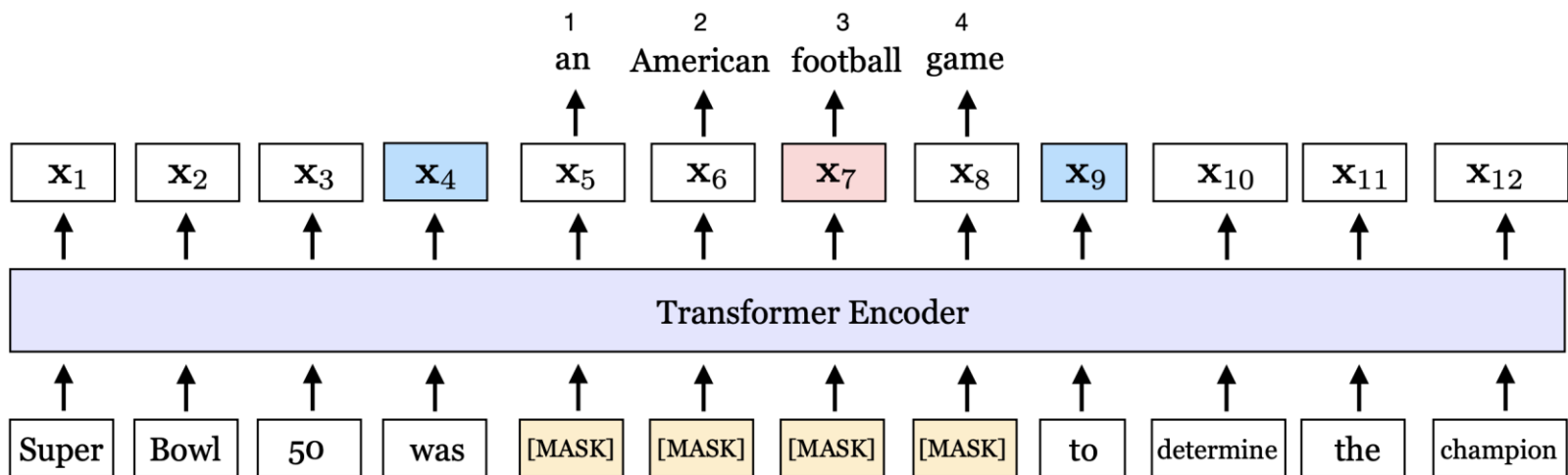
# Stanford CS224W: Masked Label Prediction

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# Masked Label Prediction

- An alternative approach to **explicitly include node label information** (works with GNN).
- Inspired from BERT objective in NLP.
  - Pretraining strategy: **masked word prediction**



# Masked Label Prediction

- **Idea: Treat labels as additional features**
  - Concatenate the **node label matrix**  $Y$  with the **node feature matrix**  $X$
- **ML setting: Use partially observed labels  $\hat{Y}$  to predict the remaining unobserved labels**
  - **Training:** First corrupt  $\hat{Y}$  into  $\tilde{Y}$  by randomly masking a portion of node labels to zeros, then use  $[X, \tilde{Y}]$  to predict the masked node labels.
  - **Inference:** Employ all  $\hat{Y}$  to predict the remaining unlabeled nodes (in the validation/test set).
  - **Similar to link prediction!** Also a self-supervised task

# Summary of the Lecture

- We discussed 3 ideas that **explicitly use labels** when making predictions on graphs
  - **Label propagation**
    - Directly propagate known labels to all the nodes
  - **Correct & Smooth**
    - First define a base predictor, then correct and smooth the predictions with label propagation
  - **Masked label prediction**
    - Construct a self-supervised ML task, let graph ML model to propagate label information

# Summary of ML with Graphs

- We discussed 3 frameworks to approach **machine learning tasks on graphs**
- **Node embeddings**
  - Embed nodes into Euclidean space, and use distance metric in embedding space to approximate node similarity
- **Graph Neural Networks**
  - Iterative neighborhood aggregation
- **Label Propagation**
  - Inductive bias: homophily
  - Explicitly incorporate label information when making predictions